Agile Parking

Projeto Arquitetural

There is guidance within this template that appears in a style named InfoBlue. This style has a hidden font attribute that allows you to toggle whether it is visible or hidden in this template. Use the Microsoft® Word® menu **Tools > Options > View > Hidden Text** check box to toggle this setting. There is also an option for printing: **Tools > Options > Print**.

# Objetivo

Descrever todo o mecanismo arquitetural como metas, requisitos e abstrações que serão utilizados durante o projeto e sua implementação.

# Metas Arquiteturais e Filosofia

O sistema deverá suportar algumas qualidades de acordo com sua natureza, tais como confiabilidade, manutenabilidade, segurança, usabilidade, dentre outros que irão variar para cada aplicação sendo desenvolvida no sistema. A escolha desse software foi realizada com base da necessidade de organização em estacionamentos viabilizando melhorar o atendimento, controlar a entrada e saída de veículos, facilitar pagamentos, e levantar estimativas de lotação.

# Premissas e Dependências

* Atender a todas as funcionalidades do sistema;
* Qualificação na plataforma escolhida;
* Habilidade nas ferramentas escolhidas;

# Requisitos Críticos da Arquitetura

* Segurança: sua falha pode proceder em prejuízos, danos ambientais e perda da vida útil.
* Missão: sua falha pode ocasionar problema em alguma atividade conduzida a metas.
* Negócio: sua falha pode resultar em custos elevados para a empresa que trabalha com o software

Para especificar um sistema crítico, é preciso compreender os riscos e gerar requisitos de confiabilidade para lidar com eles. No entanto, é preciso saber identificar os riscos e aplicar a cada tipo de sistema. Um sistema considerado crítico, precisa ser de total confiança. Há diversas razões para que a confiança seja de fundamental importância, tais como:

* Sistemas não confiáveis ou desprotegidos: se não confiam nele, com certeza não vão querer utilizá-los;
* Custos de falhas elevados: os custos de falhas no software são altíssimos;
* Sistemas que levam a desconfiança ocasionam perdas de informações: as perdas de informações são ocorridas nesses softwares e podem ocasionar prejuízos enormes.

Devido ao alto custo nas falhas dos sistemas, eles são, na maioria das vezes, desenvolvidos com técnicas já conhecidas. Essas técnicas, embora antigas, tem seu ponto forte e fraco já conhecido, mas, ainda há componentes nos quais as falhas podem ocorrer, como exemplo: o hardware do software pode falhar; podem ocorrer erros na especificação, projeto ou implementação e falhas humanas na operação do software.

A extração dos requisitos será feita à medida que os casos de uso do sistema são realizados e validados.

# Decisões, Restrições e Justificativas

* **Implementação de ferramentas auxiliares:** Serão utilizadas ferramentas como o primefaces para facilitar no designer e comunicação do sistema web.
* **Padrão da Arquitetura:** Utilização de padrão MVC, para facilitar na modificabilidade, reusabilidade e extensibilidade.

# Mecanismos Arquiteturais

**Trabalhar com JSF:**

O Java ServerFaces é uma das mais atuais alternativas para se construir a camada view da aplicação web. Ela permite ao desenvolvedor utilizar tags específicas, sem nenhuma codificação nativa em Java (scriptles) no arquivo jsp. As páginas dinâmicas ficam ligadas diretamente aos Beans. A produtividade é favorecida e a manutenção facilitada, como veremos.



**Padrão JSF com MVC:**

No JavaServer Faces, o controle fica por conta de um servlet chamado Faces Servlet, por arquivos de configuração (ex.: faces-config.xml), pelos Backing Beans e pelos validadores e conversores. O Faces Servlet se designa a receber requisições, encaminhá-las ao modelo e enviar as respectivas repostas. Os arquivos de configuração são responsáveis por definirem a navegação entre páginas e o mapeamento de ações. Já os validadores e conversores permitem um maior controle sobre os dados que serão enviados. Importante ressaltar que, mesmo sendo referenciados nas páginas, os conversores e validadores fazem parte do processamento de eventos, logo também fazem parte do controlador e não da visão. O modelo é constituído por classes de entidade e de negócio, que por sua vez, recebem dados da camada de visualização e executam regras pertinentes ao negócio. Já a visualização, remete ao que o usuário vê, ou seja: a interface - páginas JSP e kits renderizadores (HTML, WML, XML, etc.). Mas, como o JSF se utiliza de componentes, pode se dizer que esta é composta por uma hierarquia de componentes organizados. De modo simplificado, o MVC em aplicações web é distribuído da seguinte forma:

**Visão**:

* Componentes UI em páginas JSP/XHTML
* Kits renderizadores (HTML, WML, XML, etc.)

**Controlador**:

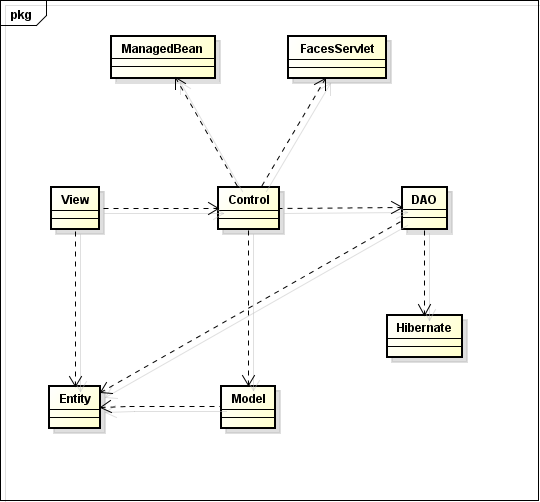
* Faces Servlet (Front Controller)
* Managed Bean (Page Controller ou Modelo)

**Modelo**:

* Entidades e regras de negócio
* Objetos gerais da aplicação (dados, etc.)



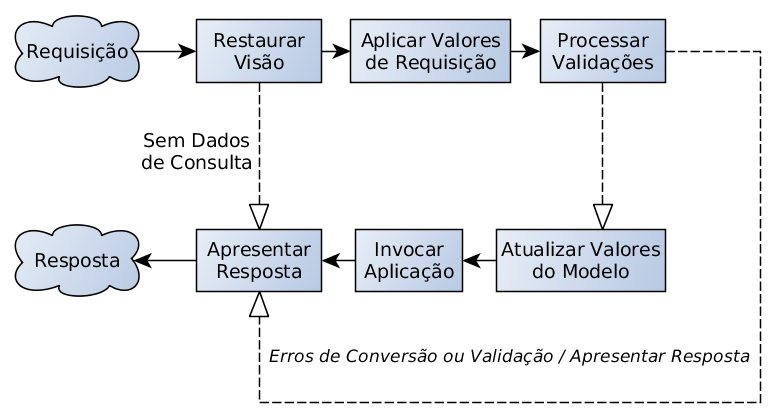
# Principais Abstrações



# Camadas do Framework da Arquitetura

O JSF, diferente de outros frameworks, possui um processamento de requisição dividido em seis fases:

1. Restauração da visão - o objetivo principal desta fase é construir a [árvores de componentes](http://pt.wikipedia.org/w/index.php?title=%C3%81rvores_de_componentes&action=edit&redlink=1" \o "Árvores de componentes (página não existe)). Ela utiliza o template e cria a árvore inicial através da análise da requisição. Após isto, ela salva o estado da árvore no objeto FacesContext. Nas requisições subsequentes, ela cria a árvore do estado e procede a execução do resto do ciclo de vida.
2. Aplicação dos valores de requisição - o JSF pega cada componente da árvore começando com a raiz e a cria ou recupera do objeto FacesContext. Cada componente na árvore gerencia seus próprios valores e toma-os dos parâmetros, cookies e cabeçalhos da requisição HTTP.
3. Validações de processo - o JSF realiza a conversão e validação sobre todos os componentes começando com o raiz. O valor submetido de cada componente é convertido em um objeto e validado chamando-se o validador registrado. O JSF salva o valor submetido. Se ocorrer um erro durante a conversão ou validação, o ciclo de vida escapa diretamente para a fase de "apresentação da resposta".
4. Atualização de valores de modelo - durante esta fase, o valor do componente é passado para o modelo através da atualização das propriedades dos backing beans.
5. Invocação da aplicação - a manipulação de evento para cada ação e ouvidor de ação é executada começando com o(s) ouvidor(es) de ação e então a chamada do método de ação.Apresentação da resposta



# Visões Arquiteturais

Caso de Uso:

